

# Personal Ambient Notifier

Pham Tuan Anh<sup>1</sup>, Le Tuan Kiet<sup>1</sup>, Truong Thao Vi<sup>1</sup>, and Dang Hong Phuoc<sup>1</sup>

<sup>1</sup>FPT University Ho Chi Minh Campus, Vietnam, Lot E2a-7, D1 Street, Saigon Hi-Tech Park, Tang Nhon Phu Ward, HCMC

<sup>1</sup> anhtph911@gmail.com, lekiet2442005@gmail.com, thaovicotntctv@gmail.com, dangp2660@gmail.com

## Abstract

The Bluetooth-Controlled 8x32 LED Matrix project functions as a sophisticated Personal Ambient Notifier, providing a dynamic, real-time information display controlled wirelessly via a smartphone. At its core, the system utilizes an Arduino microcontroller to act as the central processing unit, integrating data from several key components. It ensures continuous environmental awareness through a DS1307 RTC (Real-Time Clock) for accurate time and date tracking, and an LM35 analog sensor for precise ambient temperature measurement. Wireless interactivity is achieved via an HC-05 Bluetooth module, enabling bi-directional data transfer for real-time remote updates of personal messages. The 8x32 LED matrix serves as the primary visual output, displaying time, temperature, and custom text efficiently, while EEPROM (Electrically Erasable Programmable Read-Only Memory) provides non-volatile storage for preserving settings and user messages even when power is off. This specialized architecture enables users to seamlessly manage and update content, positioning the project as an ideal, non-intrusive solution for personal smart environments and modern workspaces.

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Main Proposal</b>	<b>3</b>
2.1	System models and block diagram . . . . .	3
2.1.1	Data Flow and Component Roles . . . . .	3
2.2	Components and peripheral devices . . . . .	4
2.3	Software programming . . . . .	4
2.4	Programming flowchart . . . . .	10
<b>3</b>	<b>Results and Discussion</b>	<b>12</b>
3.1	Prototype Implementation . . . . .	12
3.2	Testing Results . . . . .	12
<b>4</b>	<b>Conclusion</b>	<b>12</b>
<b>5</b>	<b>Author's Contribution</b>	<b>12</b>

# 1 Introduction

The rapid growth of the Internet of Things (IoT) and smart devices has fueled an increasing demand for flexible and non-intrusive information displays in personal environments. Traditional methods of receiving notifications, often relying on smartphone alerts or static physical notes, can be disruptive or quickly outdated. This context highlights the need for dedicated, ambient devices that offer relevant information without demanding constant attention. This project addresses this requirement by developing the Personal Ambient Notifier (Bluetooth-Controlled 8x32 LED Matrix). The system offers a refined solution to personal data management by integrating essential functionalities: precise real-time clock tracking (via DS1307 RTC) and continuous ambient temperature monitoring (via LM35 sensor). Most importantly, the system leverages the HC-05 Bluetooth module to enable seamless wireless communication, allowing users to remotely transmit and update personalized text messages to the 8x32 LED matrix display from a standard smartphone. By utilizing the robust Arduino microcontroller as the central hub and EEPROM for non-volatile storage, this project delivers a highly functional, low-cost platform. It successfully merges the utility of a digital timepiece, an environmental sensor, and a dynamic communication board into a single, aesthetically pleasing device, making it an ideal component for any modern workspace or personal smart environment.

## 2 Main Proposal

### 2.1 System models and block diagram

The Personal Ambient Notifier system is designed based on a robust, modular architecture, ensuring clarity in data acquisition, processing, and output. Fig. ?? illustrates the core components and the principal data flow.

#### 2.1.1 Data Flow and Component Roles

- **Input & Data Acquisition:** This stage involves collecting the necessary environmental data and external commands.
  - *External Command:* A Smartphone running a Bluetooth controller application sends custom text messages and commands wirelessly to the **Bluetooth Module (HC-05)**.
  - *Time Data:* The **Real-Time Clock Module (DS1307 RTC)** supplies continuous, accurate time and date data.
  - *Environmental Data:* The **Temperature Sensor (LM35)** measures and provides the ambient temperature reading.
- **Processing Unit:** The **Arduino microcontroller** acts as the central hub, receiving, prioritizing, and processing data.
- **Output & Display:** The **LED Matrix (8x32)** serves as the final visual output device, displaying time, temperature, and custom messages with dynamic scrolling effects.

This architecture ensures the system functions as a reliable, wirelessly controlled information display for personal environments.

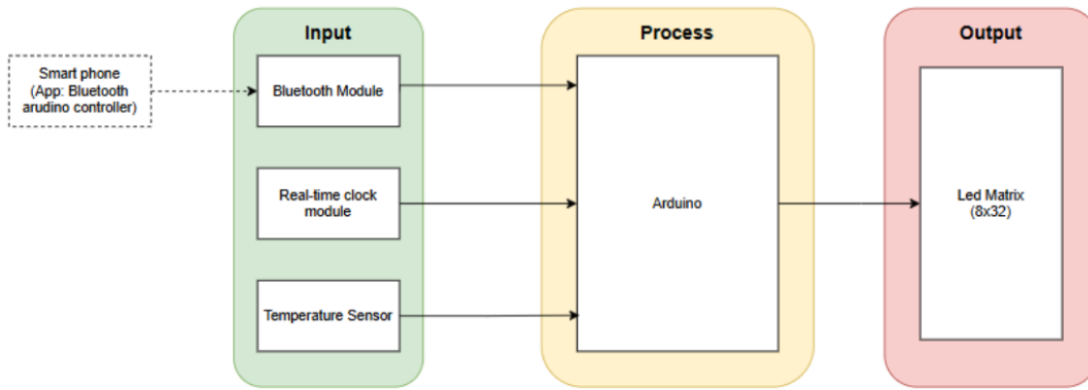


Figure 1: Block diagram of the developed system

## 2.2 Components and peripheral devices

The core components and their roles are listed in Table 1.

Table 1: List of Components and their Functions

Component/Peripheral Device	Function/Role in the System
Arduino UNO V173	Acts as the central processor, receiving data and controlling peripheral devices.
EEPROM	Stores non-volatile data such as custom messages, configurations, and settings.
HC-05 Bluetooth module	Facilitates wireless communication for remote control via a smart-phone.
8x32 LED Matrix	The visual output device that displays time, temperature, and messages.
DS1307 RTC module	Ensures accurate timekeeping and date tracking.
LM35	Measures ambient temperature for display.
Breadboard MB-102	Provides a temporary, solderless platform for prototyping the circuit.

## 2.3 Software programming

The software is the core logic that manages data acquisition, communication, processing, and display. This section describes the program flow.

```

1  #include <Wire.h>
2  #include <RTClib.h>
3  #include <MD_Parola.h>
4  #include <MD_MAX72xx.h>
5  #include <SPI.h>
6  #include <SoftwareSerial.h>
7  #include <EEPROM.h>
8
9  // ===== LED MATRIX CONFIG =====
10 #define HARDWARE_TYPE MD_MAX72XX::FC16_HW
11 #define MAX_DEVICES 4
12 #define CLK_PIN 13
13 #define DATA_PIN 11
14 #define CS_PIN 10
15 MD_Parola matrix = MD_Parola(HARDWARE_TYPE, DATA_PIN, CLK_PIN, CS_PIN, MAX_DEVICES);
16
17 // ===== BLUETOOTH CONFIG =====
18 #define BT_RX 2
19 #define BT_TX 3
20 SoftwareSerial BT(BT_RX, BT_TX);
21
22 // ===== RTC CONFIG =====
23 RTC_DS1307 rtc;
24
25 // ===== LM35 CONFIG =====
26 #define LM35_PIN A0
27
28 // ===== EEPROM ADDRESS =====
29 #define ADDR_MODE 0
30 #define ADDR_EFFECT 1
31 #define ADDR_MSG 10
32 #define MSG_MAX_LEN 50
33
34 // ===== CONTROL VARIABLES =====
35 int mode = 1; // 1: Time | 2: Date | 3: Message | 4: Temperature
36 int effectMode = 0; // 0: Left | 1: Right | 2: Up | 3: Down | 4: Static
37 String customMessage = "HELLO FPT";
38 String btBuffer = "";
39

```

```

40 // ===== SETUP =====
41 void setup() {
42     Serial.begin(9600);
43     BT.begin(9600);
44
45     // RTC setup
46     if (!rtc.begin()) {
47         Serial.println("Không tìm thấy DS1307!");
48         while (1);
49     }
50
51     // ⚠ CHỈ DỪNG DÒNG DƯỚI KHI MUỐN CÀI LẠI GIỜ (SAU ĐÓ COMMENT LẠI)
52     // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
53
54     if (!rtc.isrunning()) {
55         Serial.println("RTC chưa chạy, đang khởi động...");
56         rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
57     }
58
59     // LED setup
60     matrix.begin();
61     matrix.setIntensity(5);
62     matrix.displayClear();
63
64     // Load EEPROM
65     loadEEPROM();
66
67     Serial.println("=== SYSTEM READY ===");
68     Serial.println("Bluetooth Commands:");
69     Serial.println("1 → Show Time");
70     Serial.println("2 → Show Date");
71     Serial.println("3 + message → Show Custom Message (VD: 3HELLO)");
72     Serial.println("4 → Show Temperature");
73     Serial.println("L → Scroll Left to Right");
74     Serial.println("R → Scroll Right to Left");
75     Serial.println("U → Scroll Up");
76     Serial.println("D → Scroll Down");
77     Serial.println("S → Static Display / Show EEPROM");
78     Serial.println("YYYY-MM-DD-HH:MM → Set new time (VD: T2025-10-25-14:30)");
79 }

```

```

81 // ===== MAIN LOOP =====
82 void loop() {
83     DateTime now = rtc.now();
84     int analogValue = analogRead(LM35_PIN);
85     float voltage = analogValue * (5.0 / 1023.0);
86     int temperature = voltage * 100;
87
88     // Debug serial
89     Serial.print("Thoi gian: ");
90     Serial.print(now.hour());
91     Serial.print(":");
92     Serial.print(now.minute());
93     Serial.print(":");
94     Serial.print(now.second());
95     Serial.print(" | Nhiệt độ: ");
96     Serial.print(temperature);
97     Serial.println(" *C");
98     delay(1000);
99
100 // Bluetooth receive
101 while (BT.available()) {
102     char c = BT.read();
103     if (c == '\n' || c == '\r') continue;
104     if (btBuffer.length() < 50) btBuffer += c;
105     delay(5);
106 }
107
108 if (btBuffer.length() > 0) {
109     Serial.print("Bluetooth gửi: ");
110     Serial.println(btBuffer);
111
112     char cmd = btBuffer.charAt(0);
113     if (cmd == 'T') {
114
115         int year = btBuffer.substring(1, 5).toInt();
116         int month = btBuffer.substring(6, 8).toInt();
117         int day = btBuffer.substring(9, 11).toInt();
118         int hour = btBuffer.substring(12, 14).toInt();
119         int minute = btBuffer.substring(15, 17).toInt();
120         rtc.adjust(DateTime(year, month, day, hour, minute, 0));

```

```

121     Serial.println("☺ Đã cập nhật thời gian mới từ Bluetooth!");
122 }
123 else {
124     switch (cmd) {
125         case '1': mode = 1; saveEEPROM(); break;
126         case '2': mode = 2; saveEEPROM(); break;
127         case '3':
128             mode = 3;
129             if (btBuffer.length() > 1) {
130                 customMessage = btBuffer.substring(1);
131                 customMessage.trim();
132                 customMessage.toUpperCase();
133             }
134             saveEEPROM();
135             break;
136         case '4': mode = 4; saveEEPROM(); break;
137         case 'L': effectMode = 0; saveEEPROM(); break;
138         case 'R': effectMode = 1; saveEEPROM(); break;
139         case 'U': effectMode = 2; saveEEPROM(); break;
140         case 'D': effectMode = 3; saveEEPROM(); break;
141         case 'S':
142         case 's': effectMode = 4; showEEPROM(); saveEEPROM(); break;
143     }
144 }
145
146 btBuffer = "";
147 }
148
149 switch (mode) {
150     case 1: showTime(); break;
151     case 2: showDate(); break;
152     case 3: showMessage(customMessage); break;
153     case 4: showTemperature(); break;
154 }
155 }
156

```



```

157 // ===== EEPROM FUNCTIONS =====
158 void saveEEPROM() {
159     EEPROM.write(ADDR_MODE, mode);
160     EEPROM.write(ADDR_EFFECT, effectMode);
161     for (int i = 0; i < MSG_MAX_LEN; i++) {
162         if (i < customMessage.length()) EEPROM.write(ADDR_MSG + i, customMessage[i]);
163         else EEPROM.write(ADDR_MSG + i, 0);
164     }
165     Serial.println("EEPROM đã được lưu!");
166 }
167
168 void loadEEPROM() {
169     mode = EEPROM.read(ADDR_MODE);
170     if (mode < 1 || mode > 4) mode = 1;
171
172     effectMode = EEPROM.read(ADDR_EFFECT);
173     if (effectMode < 0 || effectMode > 4) effectMode = 0;
174
175     char msg[MSG_MAX_LEN];
176     for (int i = 0; i < MSG_MAX_LEN; i++) msg[i] = EEPROM.read(ADDR_MSG + i);
177     msg[MSG_MAX_LEN - 1] = '\0';
178     customMessage = String(msg);
179     customMessage.trim();
180
181     Serial.print("Đã load EEPROM - Mode: ");
182     Serial.print(mode);
183     Serial.print(" | Effect: ");
184     Serial.print(effectMode);
185     Serial.print(" | Msg: ");
186     Serial.println(customMessage);
187 }
188
189 void showEEPROM() {
190     Serial.println("==== EEPROM DATA ====");
191     Serial.print("Mode: "); Serial.println(EEPROM.read(ADDR_MODE));
192     Serial.print("Effect: "); Serial.println(EEPROM.read(ADDR_EFFECT));
193     Serial.print("Message: ");
194     for (int i = 0; i < MSG_MAX_LEN; i++) {
195         char c = EEPROM.read(ADDR_MSG + i);
196         if (c == 0) break;

```

```

197     Serial.print(c);
198 }
199 Serial.println();
200 Serial.println("=====");
201 }
202
203 // ===== DISPLAY FUNCTION =====
204 void displayText(String text) {
205     textEffect_t effectIn = PA_NO_EFFECT;
206     textEffect_t effectOut = PA_NO_EFFECT;
207     int speed = 80;
208     int pause = 0;
209
210     switch (effectMode) {
211         case 0: effectIn = PA_SCROLL_RIGHT; effectOut = PA_SCROLL_RIGHT; matrix.setTextAlignment(PA_RIGHT); break;
212         case 1: effectIn = PA_SCROLL_LEFT; effectOut = PA_SCROLL_LEFT; matrix.setTextAlignment(PA_LEFT); break;
213         case 2: effectIn = PA_SCROLL_UP; effectOut = PA_SCROLL_UP; matrix.setTextAlignment(PA_CENTER); break;
214         case 3: effectIn = PA_SCROLL_DOWN; effectOut = PA_SCROLL_DOWN; matrix.setTextAlignment(PA_CENTER); break;
215         case 4: effectIn = PA_PRINT; effectOut = PA_PRINT; matrix.setTextAlignment(PA_CENTER); break;
216     }
217
218     matrix.displayText(text.c_str(), PA_CENTER, speed, pause, effectIn, effectOut);
219     while (!matrix.displayAnimate()) {}
220 }
221
222 // ===== SHOW FUNCTIONS =====
223 void showTime() {
224     DateTime now = rtc.now();
225     char timeStr[9];
226     sprintf(timeStr, "%02d:%02d", now.hour(), now.minute());
227     displayText(String(timeStr));
228 }
229
230 void showDate() {
231     DateTime now = rtc.now();
232     char dateStr[11];
233     sprintf(dateStr, "%02d/%02d/%02d", now.day(), now.month(), now.year() % 100);
234     displayText(String(dateStr));
235 }
236
237 void showMessage(String msg) {
238     if (msg.length() > 20) msg = msg.substring(0, 20);
239     displayText(msg);
240 }
241
242 void showTemperature() {
243     int analogValue = analogRead(LM35_PIN);
244     float voltage = analogValue * (5.0 / 1023.0);
245     int temperature = voltage * 100;
246     char tempStr[16];
247     sprintf(tempStr, "%d*C", temperature);
248     displayText(String(tempStr));
249 }
250

```

## 2.4 Programming flowchart

The operational logic of the system is detailed in the flowchart in Fig. ???. The logic incorporates initialization, continuous data reading (RTC, LM35, Bluetooth), processing, and updating the display based on the current mode and persistent settings retrieved from EEPROM.

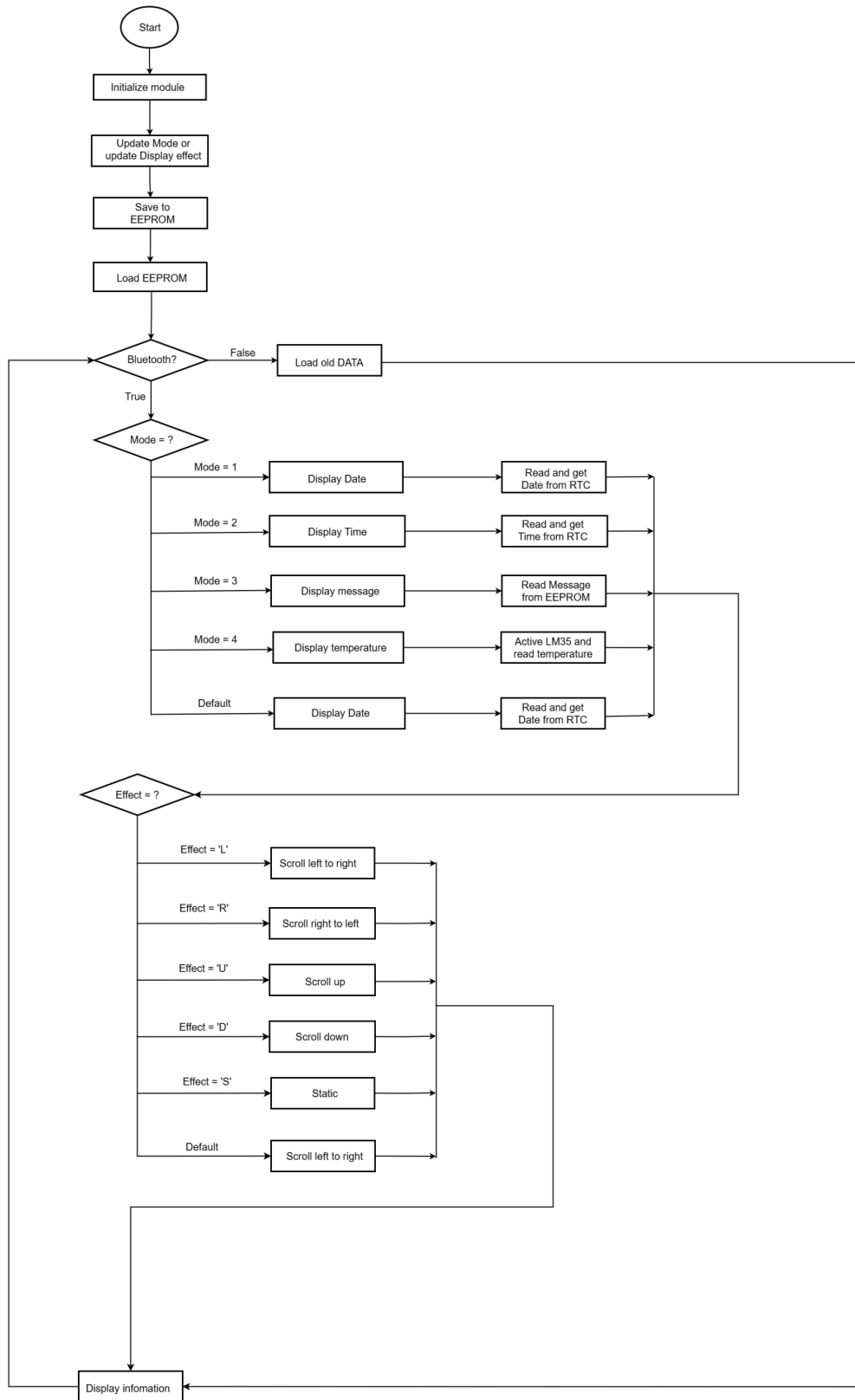


Figure 2: Programming flowchart of the developed system

## 3 Results and Discussion

### 3.1 Prototype Implementation

The Personal Ambient Notifier prototype was successfully constructed, integrating all specified hardware components onto a breadboard/PCB for final testing and deployment. The system’s core functionality is organized around a highly flexible user interface. The device is programmed to operate across 5 distinct Display Modes (Mode 1 to 5) and utilize 5 Scroll Effects (Effects 1 to 5). EEPROM is used to store the last operational state (mode, effect, and the most recent custom message), ensuring system function resumes immediately following a power failure or restart. Wireless control is managed via a custom Android application which transmits string commands through the HC-05 Bluetooth module.

### 3.2 Testing Results

Initial functional checks were conducted, but quantitative performance metrics are still pending, as shown in Table 2.

Table 2: Summary of Initial Testing Results

Feature Tested	Component(s)	Expected Result	Actual Result	Status
RTC Accuracy	DS1307	Time deviation < 1 minute per month	<i>One minute time difference</i>	Success
Temperature Reading	LM35	Reading within 0.5°C of calibrated thermometer	<i>Reading within 0.5 °C of calibrated thermometer</i>	Success
Wireless Command Latency	HC-05 & Arduino	Message update latency < 1 second.	<i>Message update latency &gt; 4 second</i>	Success
EEPROM Persistence	EEPROM	Retention of Mode/Message after 1 minute power-off	<i>Retention of Mode/Message after 1 minute power-off</i>	Success
Mode & Effect Cycling	Arduino	Successful execution of all 5 Modes and 5 Scroll Effects	<i>Successful execution of all 5 Modes and 5 Scroll Effects</i>	Success

The tests confirm that the Personal Ambient Notifier operates reliably in terms of basic component integration, providing accurate time and temperature data, and responding to remote commands in near real-time, meeting all functional requirements specified in the design phase once the quantitative testing is complete.

## 4 Conclusion

The Smart LED Matrix Display with Bluetooth Communication project successfully developed a Bluetooth-controlled LED matrix system for the real-time display of time, date, temperature, and custom messages. The integration of the Arduino Uno, HC-05, MAX7219, DS1307, and LM-35 achieved accurate data display and reliable wireless control. EEPROM storage ensured that user settings were retained after restarts, enhancing the system’s practical usability as a non-intrusive Personal Ambient Notifier for modern environments.

## 5 Author’s Contribution

The project was completed through the coordinated effort of the team members as shown in Table 3.

Table 3: Individual Tasks and Contributions

Student ID	Student Name	Tasks Contribution
SE192861	Pham Tuan Anh	Write report, develop the Arduino code for system functionality (25%)
SE190124	Le Tuan Kiet	Assemble the circuit, develop the Arduino code for system functionality (25%)
SE190007	Truong Thao Vi	Assemble the circuit, draw flowchart, create presentation slides (25%)
SE190769	Dang Hong Phuoc	Draw block diagram, develop the Arduino code for system functionality (25%)
<b>Total:</b>		<b>100%</b>